UNIVERSIDAD TECNOLÓGICA CENTROAMERICANA

UNITEC

ASIGNATURA: SISTEMAS INTELIGENTES

SECCIÓN: 1296

Tarea #3 – Arboles y Bosques

PRESENTADO POR:

DANIEL AUGUSTO MORALES ALVARADO CTA: 11941247

JUDÁ AARON PONCE VILLALTA CTA: 11841248

CATEDRÁTICO: Dr. KENNY M. DÁVILA

CAMPUS TEGUCIGALPA 20 DE NOVIEMBRE DE 2022

Índice

Introducción 1
Implementación 2
Resultados 3
Análisis 6
3 atributos más importantes 6
3 atributos menos importantes 6
Dificultades Encontradas 7
Análisis del efecto de la profundidad del árbol 7
Análisis del efecto del tamaño del dataset 8
Análisis del efecto del criterio utilizado 8
Sorpresas 9
Planteamiento 9
Conclusiones 11

Introducción

El presente reporte evidenciará la aplicación de árboles de decisión y random forest, cada uno con sus propios atributos. Se analizará el comportamiento de entrenar un solo árbol como un bosque con n cantidad de árboles. Como sabemos un árbol decisiones se asemeja mucho a la vida diaria ya que depende de las decesiones que uno tome este traerá consecuencias buenas o malas. En este caso se entrenarán arboles con 4 tamaños distintos de detataset, así como también su criterio y profundidad. Al finalizar el entrenamiento con los diferentes parámetros se tendrán 32 combinaciones en total y esto datos serán escritos en un archivo CSV, por otro lado, el random forest se entrenará con los mismos 4 datasets, en cambio se reemplaza criterio por n cantidad de árboles para la evaluación de los resultados, mismos que serán escritos en un archivo CSV también y se tendrán 48 combinaciones en total.

Cuando hablamos de entrenamiento de estas estructuras no debemos de descartar el rendimiento de cada una de estas, es por ello que se calcularán diferentes tipos de métricas tomando en cuenta los atributos y el dataset con el que se va a entrenar, scikit-learn provee las funciones para hacer estos cálculos, todos estos serán guardados en un archivo CSV indicando el dataset, los atributos utilizados y promedios de entrenamiento y validación, también el tiempo que tardará en entrenar como en validar. Todos estos resultados serán analizados en este reporte.

En definitiva, este tipo de trabajos o experimentos son necesarios para comprender el proceso que lleva entrenar este tipo de estructuras considerando que entre mas datos haya disponibles para entrenar, los resultados serán mejores y más fáciles de analizar, con pocos datos es muy difícil poder sacar conclusiones claras y las futuras decisiones que tomaran posterior a esto.

Implementación

Para llevar a cabo esta tarea se realizaron 5 script distintos, 3 de estos solicitados previamente y 2 extra que tienen la misma funcionalidad que el script 1 y 2 con diferencia que realizan el entrenamiento de forma automática con las diferentes combinaciones.

Se utilizaron 6 librerías en total que son las siguientes:

- sys
- pandas
- time
- pickle
- scikit-learn
- matplotlib

Para poder ejecutar el script 1 relacionado con decision tree, basta con colocar el siguiente comando: py .\decision_tree.py [dataset de entrenamiento] [criterion] [Depth] con el que se quiere entrenar criterio profundidad, por ejemplo: py.\decision_tree.py .\training_data_small.csv gini 2.

Hay que tomar en cuanta que tanto el dataset, criterio y profundidad son variables, es decir que por cada vez que se quiera entrenar esos parámetros pueden cambiar, para incluir la opción “None” basta con obviar la profundidad, por ejemplo: py.\decision_tree.py .\training_data_small.csv gini. Para el caso de random forest solo se cambia el tipo de entrenamiento y criterio por n árboles, por ejemplo: py .\random_forest.py .\training_data_small.csv 10 2.

Para el script 3 se coloca el siguiente comando:

py .\model_accuracy_validator.py .\Modelos\RandomForest\RandomForest-very-100-4.pkl .\validation_data.csv, (py .\model_accuracy_validator.py [modelo entrenado] [datos de validación]) aquí se debe de acceder a la carpeta modelos que contiene tanto los árboles de decisión como los random forest ya entrenados.

Los nombres de los archivos entrenados tienen la siguiente nomenclatura: RandomForest-datasetSize-n_estimators-depth.pkl, y DecisionTree-datasetSize-criterio-depth.pkl.

Para los 2 script extra solo es necesario el siguiente comando: py .\scriptAutomatizadoDecision.py o py .\scriptAutomatizadoRandom.py, que como se menciona al inicio, entrena de manera automática todas las combinaciones solicitadas.

Resultados

Para guardar los resultados tanto de decision tree y random forest, se crearon archivos CSV dentro del script 3 que permite la creación y escritura o solo escritura para los modelos que se vayan validando y que se guarden el archivo correcto y por orden, es decir por size, criterio-depth para decision tree y size nTree-depth para random forest, y cada calculo delimitado por un coma (,).

Los resultados que se obtuvieron fueron los siguientes:

Train Datase	Criterion	Depth	Train Acc.	Val Acc.	Val. Avg Rec	Val Avg. Prec	Val Avg. F1	Time Train	Time Val.
small	entropy	2	0.52	0.38	0.38	0.3863	0.3677	0.00951	0.001003
small	entropy	4	0.67	0.4275	0.4275	0.4437	0.4303	0.005508	0.001504
small	entropy	8	0.95	0.36	0.36	0.3612	0.3603	0.005547	0.001001
small	entropy		0.98	0.355	0.355	0.3576	0.356	0.006523	0.001003
small	gini	2	0.55	0.4525	0.4525	0.4649	0.4493	0.017054	0.001008
small	gini	4	0.67	0.4075	0.4075	0.4185	0.4076	0.007514	0.00208
small	gini	8	0.93	0.3925	0.3925	0.3938	0.3911	0.006548	0.001038
small	gini		0.98	0.3725	0.3725	0.3727	0.3722	0.010546	0.000966
medium	entropy	2	0.475	0.3925	0.3925	0.3031	0.3408	0.007485	0.001001
medium	entropy	4	0.585	0.445	0.445	0.4397	0.4374	0.005541	0.001033
medium	entropy	8	0.89	0.44	0.44	0.4433	0.4365	0.009546	0.001001
medium	entropy		0.985	0.435	0.435	0.4377	0.4353	0.008558	0.001001
medium	gini	2	0.475	0.3925	0.3925	0.3031	0.3408	0.01608	0.000965
medium	gini	4	0.595	0.47	0.47	0.486	0.4416	0.009521	0.001996
medium	gini	8	0.885	0.455	0.455	0.4564	0.452	0.006274	0.001505
medium	gini		0.985	0.4325	0.4325	0.4337	0.4322	0.009516	0.000993
large	entropy	2	0.435	0.4275	0.4275	0.3574	0.3717	0.011525	0.001522
large	entropy	4	0.52	0.4825	0.4825	0.4891	0.4807	0.010737	0.002524
large	entropy	8	0.775	0.4225	0.4225	0.417	0.4154	0.009558	0.001
large	entropy		0.965	0.4225	0.4225	0.4184	0.4159	0.007548	0.002006
large	gini	2	0.4225	0.3925	0.3925	0.445	0.3752	0.016956	0.001037
large	gini	4	0.535	0.5	0.5	0.5103	0.5036	0.009522	0.001
large	gini	8	0.7875	0.48	0.48	0.4772	0.4776	0.006983	0.003016
large	gini		0.965	0.445	0.445	0.4463	0.4435	0.007519	0.002008
Very-Large	entropy	2	0.46	0.44	0.44	0.3338	0.3781	0.018022	0.001001
Very-Large	entropy	4	0.517	0.47	0.47	0.4725	0.4701	0.021043	0.001
Very-Large	entropy	8	0.6615	0.5125	0.5125	0.5101	0.5022	0.023062	0.00107
Very-Large	entropy		0.885	0.44	0.44	0.4453	0.4415	0.023028	0.001
Very-Large	gini	2	0.46	0.44	0.44	0.3338	0.3781	0.024438	0.001001
Very-Large	gini	4	0.5275	0.49	0.49	0.4953	0.488	0.020051	0.001996
Very-Large	gini	8	0.6625	0.4925	0.4925	0.4865	0.4827	0.022015	0.001033
Very-Large	gini		0.885	0.42	0.42	0.4234	0.4198	0.02456	0.001

Ilustración 1 Resultados para decision tree (32 combinaciones)

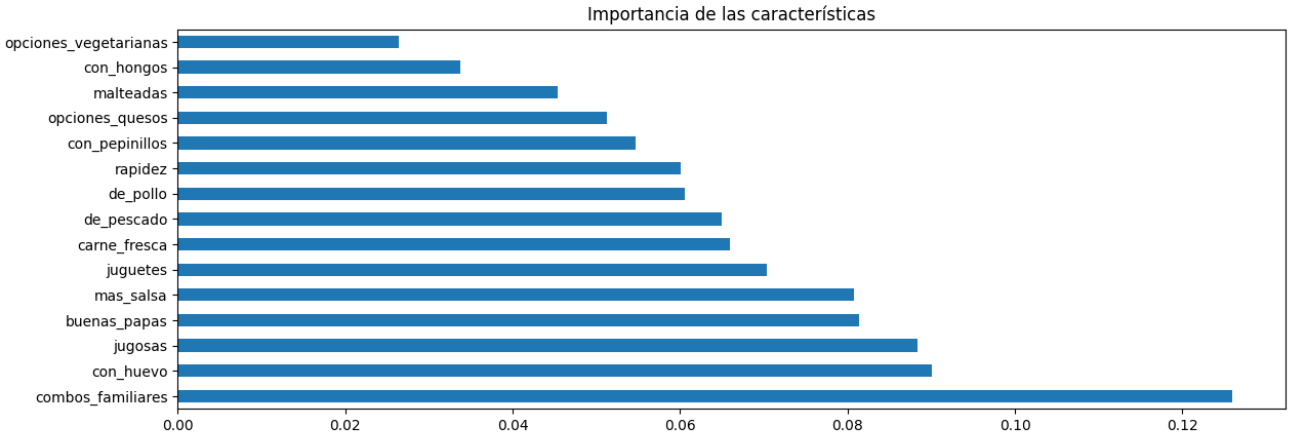
Train Dataset	N Trees	Depth	Train Acc	Val Acc.	Val. Avg Rec	Val Avg. Prec	Val Avg. F1	Time Train	Time Val.
small	5	2	0.57	0.41	0.41	0.398	0.4004	0.01217	0.002
small	5	4	0.68	0.4675	0.4675	0.4604	0.4546	0.010515	0.003
small	5	6	0.87	0.3475	0.3475	0.3605	0.3476	0.010508	0.0019
small	10	2	0.65	0.4125	0.4125	0.4102	0.4029	0.015551	0.002
small	10	4	0.78	0.4325	0.4325	0.4359	0.4329	0.015516	0.0027
small	10	6	0.91	0.3675	0.3675	0.3689	0.366	0.016364	0.0025
small	50	2	0.66	0.445	0.445	0.448	0.4421	0.055083	0.0055
small	50	4	0.86	0.4325	0.4325	0.4422	0.4315	0.052402	0.006
small	50	6	0.95	0.415	0.415	0.4345	0.4161	0.052084	0.007
small	100	2	0.67	0.4925	0.4925	0.4818	0.4775	0.091711	0.0096
small	100	4	0.81	0.4675	0.4675	0.4678	0.4661	0.090059	0.0112
small	100	6	0.95	0.4325	0.4325	0.4388	0.4309	0.09764	0.0116
medium	5	2	0.525	0.445	0.445	0.4577	0.4331	0.013028	0.003
medium	5	4	0.63	0.46	0.46	0.4614	0.4531	0.011543	0.0026
medium	5	6	0.78	0.4725	0.4725	0.4753	0.4708	0.012072	0.0025
medium	10	2	0.56	0.475	0.475	0.4708	0.4674	0.017005	0.002
medium	10	4	0.61	0.5	0.5	0.4964	0.4974	0.015564	0.002
medium	10	6	0.81	0.5	0.5	0.5011	0.4954	0.018022	0.003
medium	50	2	0.58	0.4675	0.4675	0.452	0.4477	0.048776	0.005
medium	50	4	0.69	0.5025	0.5025	0.4966	0.4945	0.053485	0.0076
medium	50	6	0.855	0.4825	0.4825	0.4781	0.4773	0.064603	0.007
medium	100	2	0.605	0.4975	0.4975	0.4954	0.4892	0.096687	0.0095
medium	100	4	0.695	0.505	0.505	0.5036	0.4962	0.121258	0.0113
medium	100	6	0.855	0.5175	0.5175	0.5149	0.5103	0.233273	0.0112
large	5	2	0.475	0.43	0.43	0.4373	0.4211	0.040585	0.002
large	5	4	0.605	0.4725	0.4725	0.4677	0.4691	0.030095	0.002
large	5	6	0.675	0.475	0.475	0.4685	0.4669	0.019019	0.002
large	10	2	0.5175	0.48	0.48	0.4693	0.4614	0.026563	0.0031
large	10	4	0.5975	0.48	0.48	0.4732	0.4735	0.031507	0.002
large	10	6	0.735	0.5125	0.5125	0.5098	0.5039	0.036593	0.003
large	50	2	0.56	0.5125	0.5125	0.5086	0.5029	0.063607	0.0055
large	50	4	0.645	0.545	0.545	0.5441	0.5391	0.059726	0.0065
large	50	6	0.7675	0.555	0.555	0.5535	0.5499	0.070913	0.0065
large	100	2	0.585	0.5175	0.5175	0.5123	0.511	0.112359	0.0105
large	100	4	0.635	0.525	0.525	0.5217	0.5192	0.106797	0.0115
large	100	6	0.77	0.5275	0.5275	0.5241	0.5215	0.135434	0.012
Very-Large	5	2	0.511	0.4625	0.4625	0.4644	0.4527	0.026543	0.002
Very-Large	5	4	0.553	0.54	0.54	0.5454	0.5363	0.025102	0.003
Very-Large	5	6	0.6125	0.5325	0.5325	0.5345	0.5303	0.025032	0.002
Very-Large	10	2	0.5405	0.51	0.51	0.5103	0.5076	0.031315	0.003
Very-Large	10	4	0.58	0.5375	0.5375	0.539	0.5336	0.032979	0.0021
Very-Large	10	6	0.605	0.545	0.545	0.5433	0.5393	0.035079	0.003
Very-Large	50	2	0.541	0.5125	0.5125	0.5115	0.5099	0.070645	0.0055
Very-Large	50	4	0.5955	0.5625	0.5625	0.5625	0.5587	0.082085	0.0055
Very-Large	50	6	0.636	0.55	0.55	0.5507	0.5489	0.083596	0.0083
Very-Large	100	2	0.5725	0.545	0.545	0.5417	0.5371	0.134702	0.0115
Very-Large	100	4	0.59	0.5575	0.5575	0.5592	0.5548	0.151673	0.0105
Very-Large	100	6	0.6385	0.5575	0.5575	0.5549	0.5546	0.147172	0.0121

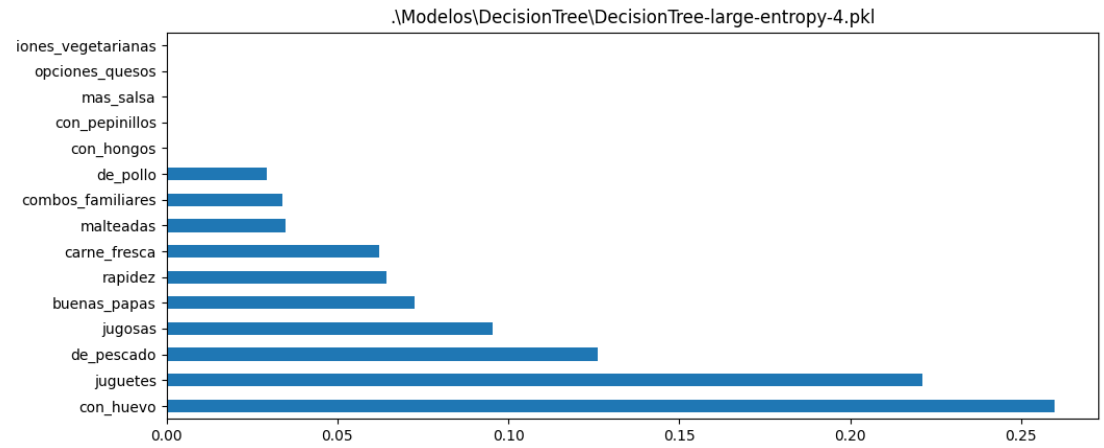
Ilustración 2 Resultados random forest (48 combinaciones)

Análisis

3 atributos más importantes

Entre todos los árboles de decisión, los atributos más recurrentes entre todas las variantes del modelo entrenadas fueron: con huevo, jugosas, y combos_familiares. Estos resultados son dados debido a que el árbol puede tomar una decisión mucho más rápido cuando se escoge alguno de estos atributos al momento de predecir, lo que nos dice que una persona que prefiera un lugar que sus hamburguesas tengan huevo, por ejemplo, es mas propenso a darnos un resultado de manera breve. En modelos donde la profundidad máxima no era especificada o era mas grande, esto se notaba un poco menos, o estos atributos se daban a notar un poco menos.





Como se ilustra en las gráficas mostradas anteriormente, también pudimos notar que opciones_vegetarianas, opciones_quesos, con_hongos y por dar una más, malteadas, fueron ilustradas como las características menos importantes para el árbol de decisión. En la vida real podríamos opinar que si un restaurante tiene opciones_vegetarianas o no podría ayudarnos a decidir más rápido, sin embargo, el algoritmo de árboles de decisión considera que estas opciones agregaban "ruido" al momento de tomar una decisión, o que simplemente llevaban a caminos que fueran mucho más largos para poder llegar a tomar una decisión en concreto.

Dificultades Encontradas

Al inicio fue confuso la realización de esta tarea, ya que estábamos validando datos donde no era. Por otro lado, para generar la tabla no podíamos acceder a todos los cálculos realizados en los otros scripts, por lo que fue necesario guardar en un objeto de tipo pickle para poder obtener y realizar las métricas solicitadas y que la mayoría de estas debían ser calculadas en el 3er script. Otra dificultad que en una parte de la documentación de scikit-learn una de las funciones para calcular las métricas estaba obsoletas y esto generaba resultados extraños. También para validar con modelos con dataset muy grandes salía un error de la división 0 ya que los valores eran continuos o infinitos por lo tanto se agregó un argumento mas que lo recomiendo scikit-learn denominado "zero division".

Análisis del efecto de la profundidad del árbol

Pudimos notar mientras analizábamos las estadísticas de los resultados que los árboles en los que no se limitaba tanto la profundidad había una mejor precisión en el entrenamiento, por otra parte, nuestros valores encontrados en la precisión de la validación fueron casi similares para todos los casos, y podemos determinar que un árbol que no tenga límite de profundidad y uno que tenga muy poca profundidad, no puede llegar a ser la mejor opción para este tipo de árboles. Así mismo, muchos atributos solían tomar los mismos valores, algunas

features no se terminaban tomando en cuenta debido a sus limitaciones, y en general los atributos o características que se determinaban como importantes eran bastante distintas a otra gran mayoría cuando no se limitaba tanto la profundidad, por lo tanto, pudimos concluir que en cierta parte al limitar mucho la profundidad de un árbol, puede llegar a tener mas ruido en cuanto a sus atributos y las decisiones que tomara.

Análisis del efecto del tamaño del dataset

Se nos fue proporcionado 4 datasets y un dataset de validación, los 4 datasets eran de diferente tamaño es decir que eran cantidades distintas de datos. Analizando el caso de overfiting el dataset que podría hacer que el modelo no fuera capaz de aprender cosas nuevas fue el dataset pequeño, ya que para que obtener resultados mas cercanos a la "realidad" se necesita de una cantidad de datos razonable, es decir que haya muchos datos. Otro factor para considerar es el sobre entrenamiento del modelo, esto tiene altas probabilidades de caer en overfitting, ya que de tanto entrenamiento y al momento de querer validar solo será capaz de reconocer aquellos datos que sean idénticos a los dataset entrenado. Es por eso que se debe de tener un equilibrio de aprendizaje o entrenamiento para los modelos. De manera general entre mas datos se tengan mejor es, entre menos datos, mas probabilidad de overfitting. En este caso el dataset que menos probabilidad de overfitting tuvo es el "very-large".

Análisis del efecto del criterio utilizado

Para los arboles de decisión los criterios utilizados fueron Gini y entropy, básicamente al utilizar Gini es que los atributos deben de pertenecer a la misma clase, quiere decir que si el índice de Gini cada vez es mayor entonces hay mas homogeneidad en los resultados. Otro detalle es que los atributos son prácticamente discretos y más fácil poder tomar decisiones. Si se piensa de otra manera, Gini mide el grado de pureza entre los atributos, entre mas grado, menos pureza, si analizamos los resultados en el dataset very large con categoría Gini,

podríamos concluir que hubo un grado bajo en el índice de Gini, es decir hubo más pureza y el porcentaje de overfitting fue relativamente pequeño, por lo tanto el modelo fue muy bien entrenado y devolvió los resultados que se esperaban Por otro lado, al utilizar entropy ya los atributos son categóricos, prácticamente no hay un orden y hay mas dificultad de poder tomar una decisión concreta, es como el inverso de Gini y una gran cantidad de datos, si analizamos los resultados al aplicar entropy con el dataset small, hay una alta probabilidad de que el entrenamiento haya sufrido overfitting y al no ser atributos con pureza, se pueden obtener datos no favorables que hagan que el árbol de decisión no tome la correcta.

Sorpresas

Pudimos notar como en la selección de los atributos o features más importantes, se tomaban en cuenta atributos que nosotros no nos imaginábamos, en nuestra opinión, un restaurante de comida rápida, tal y como lo dice el nombre debe de contar con una preparación rápida de la comida, y muchas veces no mirábamos este atributo reflejado entre los más importantes, si no que entre los menos/medio importantes en la mayoría de los árboles comparados.

Planteamiento

Estuvimos analizando varias opciones de nuevos planteamientos en lo que se podría aplicar lo aprendido, y pudimos llegar a una que nos llamaba la atención debido a la diversidad de opciones que se podrían llegar a dar. Y es un clasificador de decisiones basado en el MOBA League of Legends, que ayude a escoger a un jugador el campeón (personaje) que utilizara en su partida, que puede llegar a decidirse dados atributos como el rol a jugar, el modo de juego, si su win rate es óptimo en la temporada actual, la línea a jugar, la orientación de runas a utilizar dados otros atributos, la composición de campeones enemigos.

También aparte de eso, existen distintos tipos de campeón dada una línea o rol, como pueden llegar a ser los tanques, duelistas, campeones a distancia, asesinos, magos, campeones con artillería, campeones que curan, campeones que otorgan escudos, campeones que potencian otros campeones, o incluso campeones que roban habilidades de otros campeones. Todos los factores mencionados anteriormente se pueden llegar a vincular desde que se escoge que rol o línea que se quiere jugar en la partida, a partir de eso las opciones se pueden llegar a ampliar aún más dada la composición que el enemigo vaya escogiendo en un momento dado (tomando en cuenta que el orden de escoger campeones usualmente es 1 campeón aliado - > 2 enemigos -> 2 aliados -> 2 enemigos-> 2 aliados -> 1 enemigo o viceversa).

Dado que este juego cuenta con una variedad de campeones muy grandes (162 campeones actualmente) muchas veces 45 segundos (duración para escoger a un campeón en la previa de una partida) no es suficiente para poder llegar a tomar en consideración todas las opciones disponibles, y muy frecuentemente uno como jugador termina escogiendo el campeón que utiliza frecuentemente, dando como resultado una decisión que muy probable no favorezca a la partida, o a la composición del equipo con el que se jugara.

Conclusiones

- Se entrenaron de la manera correcta y deseada arboles y bosques con los atributos solicitados y el total de combinaciones esperadas. La librería scikit-learn es demasiado útil, su documentación es clara y muy precisa y ayudó para la finalización de los entrenamientos.

- Se almacenaron los modelos entrenados en formato .pickle, nos ayudó bastante y permitió guardar el modelo completo y con sus atributos para que estos fueran fáciles de extraer en otros scripts para calcular las métricas de dicho modelo.

- Se comprendió el funcionamiento y el proceso que lleva el entrenamiento, también se analizaron los resultados de cada modelo mismo que ayudaran en gran manera a predecir en este caso el gusto particular de los restaurantes.